RAWDATA Project portfolio

This is the **general introduction** to the **RAWDATA Project portfolio** with a brief description of the problem, the domain, (some of) the required challenges and with indications on optional directions. The portfolio will include four subprojects and the descriptions of these subprojects will be published later on the course website. **Don't start working** on the project now. **Wait until the Portfolio project 1 description is published** (around mid September).

The common goal of the project portfolio is to provide a tool to help computer programmers develop skills while they are working. The tool should support two complementary functions – a keyword-based search and a search history that keeps track of what's already retrieved and what parts were the most interesting.

The project portfolio includes four subprojects that will relate to areas covered in four corresponding sections of the RAWDATA course. The four subprojects are each supposed to be submitted before the next will be started. The general idea of the project portfolio and the overall requirements that constrain this are presented below, while more detailed requirements on the individual subprojects will be published later on the course web site. The four subprojects are to be handed in before separate deadlines during the semester, while, to finish the project portfolio, in addition a so-called *reflective synopsis* has to be submitted with deadline by the end of the semester. The reflective synopsis summarizes what is covered in the four subprojects, discuss and present reflections that relate to these and to the combined project as a whole. The synopsis should also provide an overall conclusion and comments on future work.

In brief the four subprojects will aim at building a data repository, developing services to access the repository and finally developing one or more applications utilizing the provided services. A key to a successful result is obviously to draw on a source of useful information within the domain – programming. For this purpose, we have chosen Stack Overflow.

What is Stack Overflow?

Stack Overflow (http://stackoverflow.com/) is a question and answer (QA) site for professional and enthusiast programmers, working together to build a library of detailed answers to, in principle, every question about programming. The main goal of the site is all about getting answers. It is not a discussion forum and there is no chitchat. If you register, you will be able to ask, answer, comment and vote on relevance on both answers and questions. As a guest, you can search the knowledge captured in the answers and comments to the more than 14 million questions. Furthermore, the user asking the question can point out the best answer. Due to the success of Stack Overflow it has now become a valuable resource of information for programmers seeking for answers.

In this project portfolio we will build a local data repository and load it with a limited selection of data from publicly available dumps from Stack Overflow.

What are we aiming at?

The task in the project portfolio is to create a Stack Overflow Viewer application (SOVA). The primary aim is to provide functionality for searching information in Stack Overflow. This involves keyword based search to access information, as well as some means of tracking of personal use, like search history, marking of found solutions, etc.

Like in most modern software development, the complete set of requirements is not defined up-front. Requirements will change and new ones will be added during the process. Some overall requirements, however, are often given in advance, and are very difficult to remove or change. The following that relate to architecture belongs to this category.

The application should be designed around a multilayer (multitier) architecture with at least three layers, presentation, service and data. The layers must be decoupled as much as possible, such that alternative implementations easily can be swapped in.

The presentation layer must be some kind of Web based responsive application, such that implemented app functionality is available on various platforms, from mobile to desktop devices.

The service layer defines the application logic. The design and implementation of the service layer must provide an interface to the presentation layer through web services.

The data layer encapsulates storage and retrieval of data. The interface provided by the data layer must define an abstraction that is independent of the actual representation of data in the data layer. The prototype should establish a data model, implement a database, and load data from a provided Stack Overflow data sample. Along with a Stack Overflow database a complementary database to store search history, markings, annotations etc. must be developed and implemented.

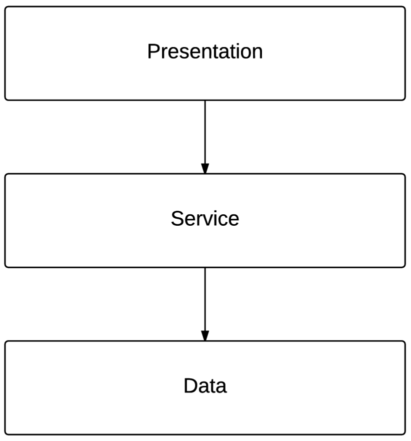

Figure 1: A multilayer architecture

Requirements and features

The tool should be developed as a responsive web application that draws on functionality made available through web services, which in turn draws on data from one or more databases and possibly other sources. The application should be designed around a multilayer architecture with at least three layers, presentation, service and data. As a minimum the application should support the following.

- Search for posts and comments in Stack Overflow.
- Present search results as (ranked) lists of posts
- Visualize search results by most frequent words using ranked lists or word clouds.
- Keep track of search history.
- Provide a marking option for posts of special interest among posts presented in the search result and allow optional annotation to marked posts.

Obviously, this list can be extended and elaborating on any of the points already there may lead to a long list of additional features. Examples of what could be included in this list are:

- Provide statistics and visualize frequent Stack Overflow topics
- Similar words search
- Phrase search
- Browse topics of interest.
- Build and visualize networks of associated words and or topics
- Provide alternative visualizations of marked/annotated posts, such as word graphs showing significant words and their relations
- …

However, the goal here is not to design and implement as many features as possible. A better way to go would be to choose a small set of features with clear motivations and clarifications for their purpose – such as they being especially helpful, nice, interesting, challenging, etc.